Lab 03

Wall Following: Feedback PD Control

Reading: *Ch. 3 of the text, Lecture 3-1*

Read this entire lab procedure before coming to lab.

****

Purpose: The purpose of this lab is to implement a wall following behavior on the GigaBot by using feedback control. The IR and/or sonar sensors will be used to detect the wall and the robot should use proportional-derivative (PD) control to maintain a distance between 4 and 6 inches from the wall. The wall following behavior should then be integrated as the top layer onto the subsumption architecture implemented in the obstacle avoidance lab.

Objectives: At the conclusion of this lab, the student should be able to:

- Acquire and use data from all of the robot's range sensors
- Implement a wall following behavior with PD control on the robot
- Use modular programming with well commented and organized code to implement subsumption architecture on the Arduino Robot
- Write a technical memo to describe the purpose, method, and results of the lab implementation

Equipment: Base robot
Sonar sensors
Lidar sensors
obstacles, walls
3 LEDs

Theory:

In this lab you will use feedback control to design a wall following behavior. The initial task will be to design a bang-bang controller that is either on or off based upon whether the robot is within an acceptable distance from the wall. If the robot is within the band it drives forward, if it is outside the band then it stops and makes one adjustment to turn back into the band. This lab builds on the prior lab except the robot now tries to maintain a given distance from the obstacle (wall) versus stopping or running away from it. After the bang-bang control is working at an acceptable level, you will design a proportional controller. Proportional control will adjust motor turn angles or speed based upon the amount of error the robot is away from the wall once it is outside of the dead band. Finally, to adjust for the rate of change or transient response of the robot movement, you will add the derivative to create a PD controller. The ambitious student team may want to also add integral control to reset the steady state error when the robot tracks the wall, but the error accumulates to a given threshold.

It is recommended that you start with a proportional controller using error based upon distance from the wall *[K_p * (error input)]*. The gain on the controller should control heading and possibly motor speed. The first step would be to tune the proportional controller by selecting

the gain with the best performance. It is recommended that the proportional controller should be adjusted until the robot follows the wall with regular oscillations or wavering but crudely maintains a distance to the wall.

Once the proportional control works at an acceptable level, try to incorporate a derivative controller, *[K_d * d (error input)/dt]*. Since the derivative of the error is the rate of change, it will be necessary to store the last value of the error and find the difference with respect to the current value and multiply by some constant. Finally, tune the derivative controller to yield the best robot performance. Devise a method to test that the wall following behavior works correctly and report the results in the lab memo. Note that reasonable gains may be between 1 and 10 and the derivative gain may be about 1/10th of the proportional gain. See the "*TuningofaPIDControllerusing Ziegler-NicholsMethod*" document in the Moodle Lab folder. Figure 1 presents a sample proportional - derivative controller for wall following.

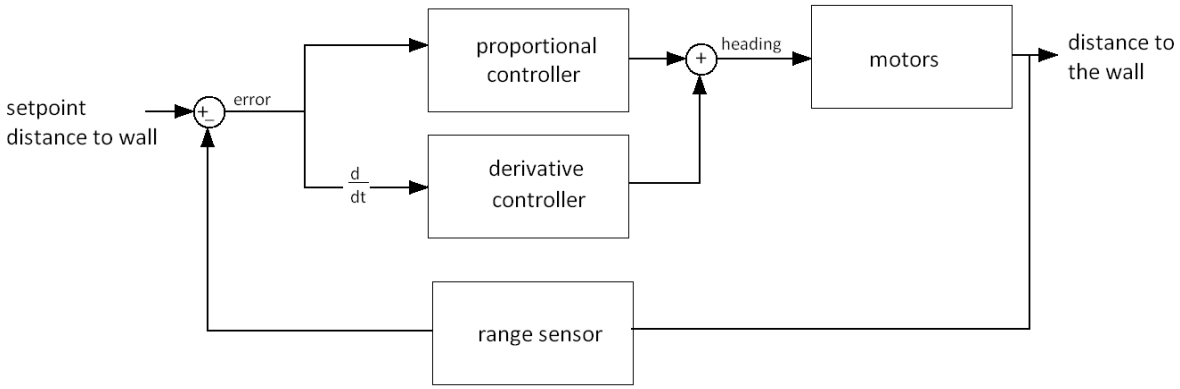

Figure 1: Wall Following PD Controller

The final code for the robot will be an integration of obstacle avoidance, random wander, wall following and hallway following. This will be implemented by using subsumption architecture and state machines as discussed in the prior lab. Note that the robot should be intelligent enough to distinguish between avoiding an obstacle and following a wall. For example, if the front distance sensor detects a wall, the collide behavior should turn on and then the robot should switch to wall following to navigate around the obstacle. Alternately, the robot can follow the wall unless the distance to the wall is less than a given distance threshold, at that point the robot should switch to avoid obstacle until it is a safe distance from the wall and then start to follow again.

The key to being able to implement this more complicated behavior is to use modularity, state machines, organization and planning at the beginning. The should be an arbitrator in your code that determines when to turn certain states on and off or to inhibit or subsume layers. See sample state machines and subsumption architectures in Figures 2-4 for guidance on how to design the robot architecture with the required layers. Note that none of these is the complete machine that you are being asked to design.

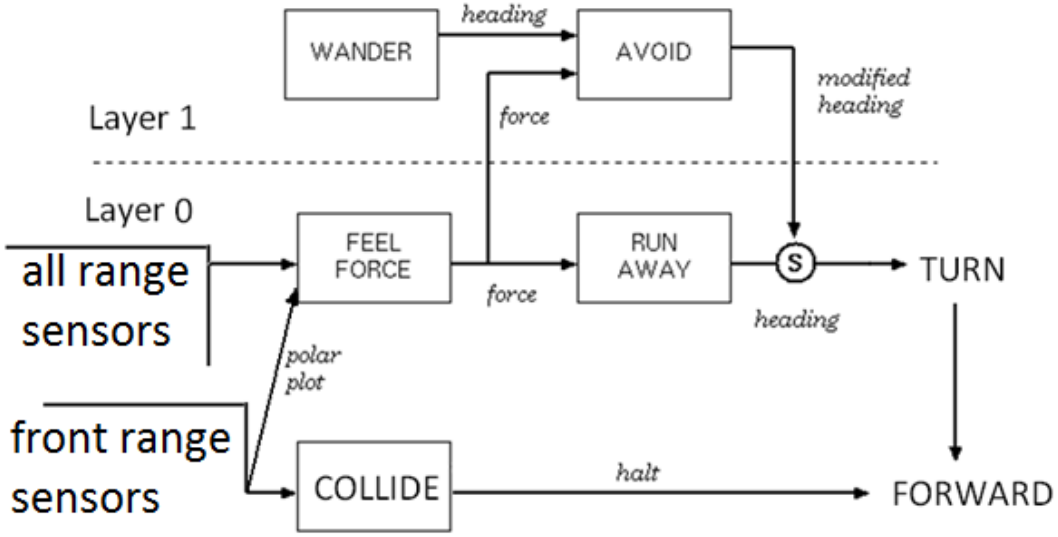

Figure 2: Subsumption Architecture (needs follow wall and hallway added to top layers)

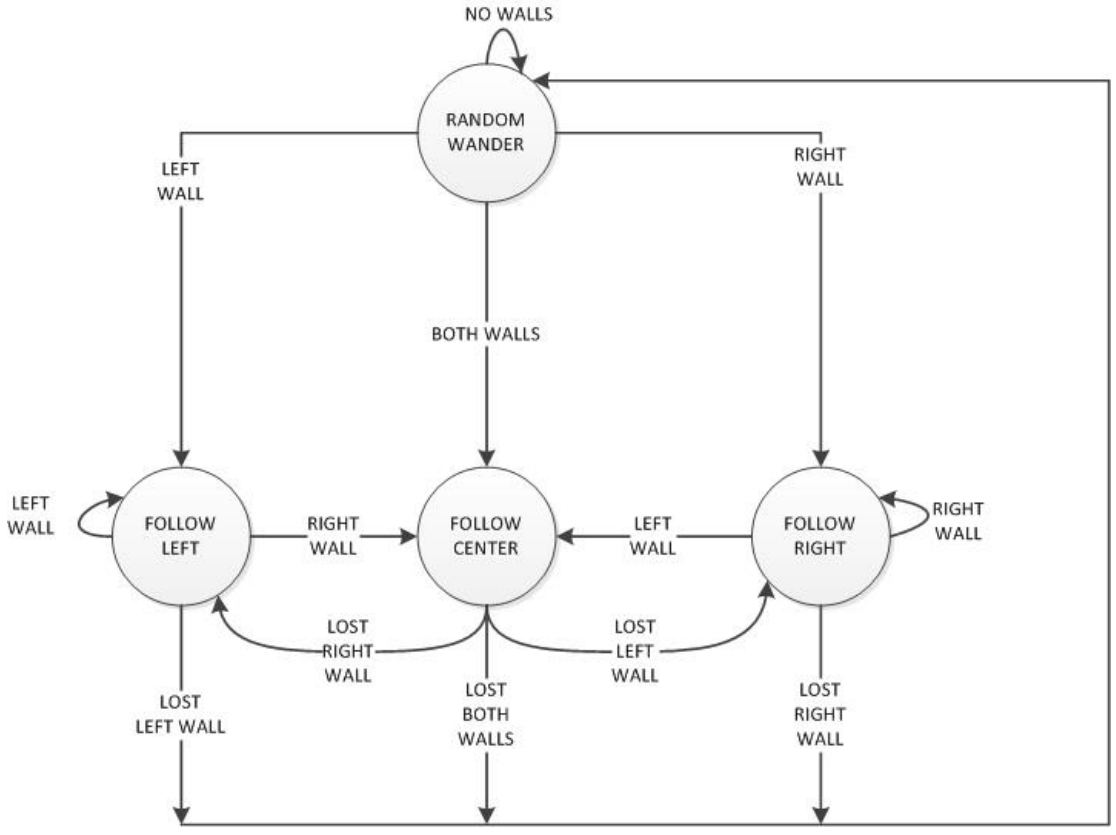

Figure 3: Wall Following State Diagram (needs an avoid obstacle state)

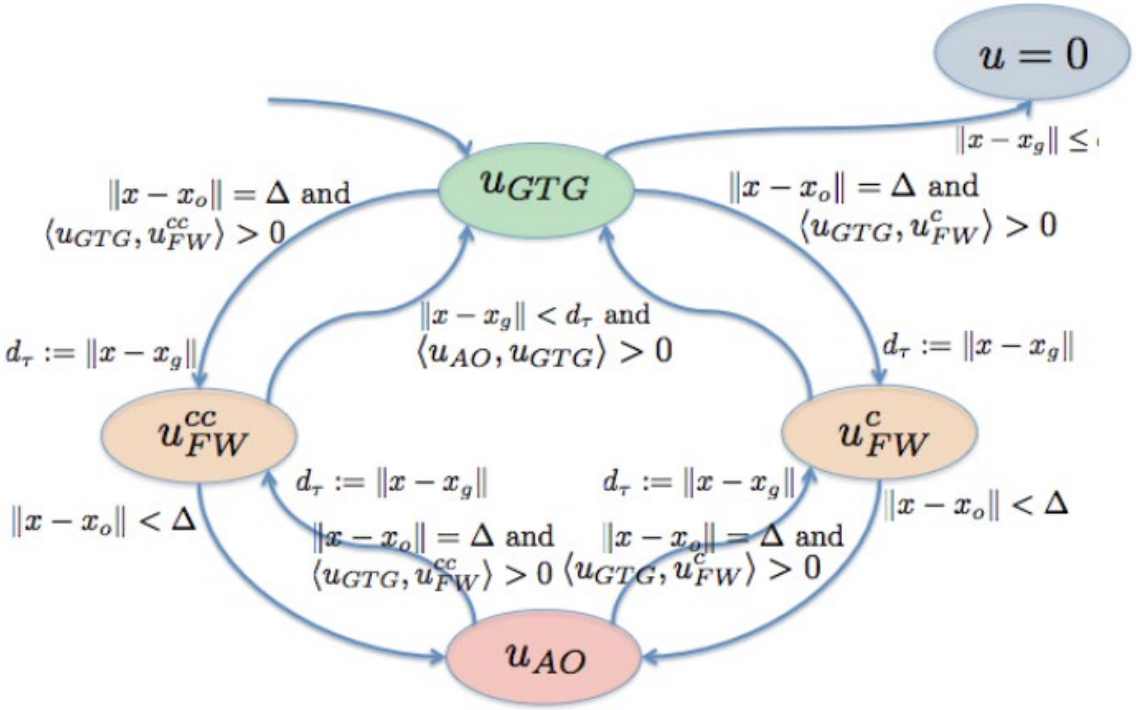

Figure 4: Go-To-Goal, Avoid-Obstacle and Follow-Wall Finite State Machine

(needs a random wander state)

Pre-Lab:

- Review the *Tuning of a PID Controller Using Ziegler-Nichols Method*" document to prepare for the lab.
- Create a state diagram with 3 states (random wander, avoid obstacle, go to goal). The inputs that transition the states are at goal, has goal, no goal, obstacle, no obstacle.
- Create the state transition table for the obstacle avoidance behavior state diagram.
- Create a subsumption architecture with random wander, avoid obstacle (runaway(too close), collide(front inside corner)), follow wall (left or right and include outside corner), and follow center behaviors.

OR

- Create the state diagram with the following states: follow wall (left or right and outside corner), follow center, random wander, avoid obstacle (run away [too close], collide [inside corner]). Inputs to the states would be based upon the range sensors.

LAB PROCEDURE

****

Part 1 – Follow Wall (Layer 1)

Bang-Bang Control

Design a wall following behavior for the robot using bang-bang control.

1. Make a copy of the code for Lab 2 and rename it "*YourRobotName-WallFollowing.ino*". You can use interrupts, threads, the multicore or loops to poll the sensors as you implement this behavior.
2. Modify the code to create a bang-bang controller. The robot should follow a wall on the left and right as long as it is within 4 to 6 inches by driving forward. If the robot moves outside of the band, the controller should turn on and make the appropriate adjustment to the motors to get the robot back within the band. The controller should be either on or off. Note that some students have found that it works better to drive the robot backward and to update sonar in the loop and IR with the timer.
3. You will eventually need a full state machine to implement your design so to keep track of the robot's states such as following on left, following on right, too far or too close from wall. The inputs to the states would be the sensor data. **Turn on the red LED when the robot is too far from the wall (outside of 6 inches). Turn on the yellow LED when the robot is too close to the wall (less than 4 inches). All LEDs should be off when the robot is between 4 and 6 inches from the wall.**
4. Once the bang-bang control is working at an acceptable level go on to the next part. Acceptable level means the robot can follow the wall for at least 2 to 3 feet by using the bang-bang control.

Proportional Control

Design a wall following behavior for the Arduino Robot using Proportional control.

1. Next, modify the code to create a proportional controller based upon the robot's distance from the wall. There should still be a dead band where the robot simply moves forward if it is within 4 to 6 inches from the wall. If it is outside of the band, the motors should be adjusted based upon the direction and magnitude of the error.
2. Once the proportional control is working at an acceptable level go on to the next part. An acceptable level means that the robot can follow the wall for at least 2 to 3 feet but there will be regular oscillations. Note that reasonable values for proportional gain may be between 1 and 10.

Proportional-Derivative Control

Design a wall following behavior for the robot using PD control.

1. Next, modify the code to create the proportional-derivative control. The derivative control should put the brakes on the robot's momentum. This value should be used to subtract off the change in the motor speed or direction based upon the rate of change in the error.
2. Adjust the gains so that the PD control is working at an acceptable level. An acceptable level means that the robot can follow the wall for at least 2 to 3 feet and the oscillations are greatly reduced. Note that reasonable values for proportional gain may be between 1 and 10 and the derivative gain may be about $1/10^{th}$ of the proportional gain.
3. Now modify the code to detect an outside corner or doorway where the robot follows the wall and makes the necessary turn to go through the doorway and hop back on the wall.
4. The state machine should be updated so that the robot keeps track of whether it was initially following left or right when the wall was lost in order to know which way to turn. **Turn on red and yellow LEDs when the robot follows the right wall. Turn on the green and yellow LEDs when the robot follows the left wall.**

Part 2 – Avoid Obstacle (Layer 0)

1. Note that when the robot gets too close to the wall it may need to switch to the runaway behavior to avoid hitting the wall and then switch back into wall following when it is at an acceptable distance. For example, if the robot is between 0 and 2 inches from the wall, execute the runaway behavior. If the robot is between 2 and 4 inches from the wall, use feedback control to move away from the wall back into the dead band of 4 to 6 inches.
2. Next, modify the code to use the front sensor to detect a front wall or corner and make the necessary adjustments to turn to navigate the corner. Note that this includes the collide behavior before switching to wall following. You should make the appropriate changes to the state machine based upon whether the robot is on the wall or in a corner. You must keep track of which wall the robot was following so that it knows how to turn to navigate the corner. *Make sure your code uses a timer or states in order to extract the robot from stuck situations.*
3. **Turn off the LEDs when the robot loses the wall or encounters an outside corner. Turn on the red and green LED when the robot encounters a front wall or an inside corner.**

Part 3 – Random Wander (Layer 3)

1. Next you will integrate the random wander created in the prior lab into the state machine.
2. Start the robot at least 10 inches from the wall or obstacles and the robot should randomly wander until it detects a wall on the left or right and then switch into wall following.
3. If the robot is following the wall and loses the wall it should attempt to hop back on the wall and after a certain number of tries if the wall is not found, then switch into random wander.
4. **Turn on the green LED when the robot is in random wander.**

Part 4 – Follow Center (Layer 2)

1. Improve the wall following behavior created in the previous part such that if the robot detects a wall on both sides (i.e. hallway), it will move to the center and stay in the middle until one of the walls is lost.
2. If one of the walls is lost, the robot should return to the basic wall following behavior.
3. If both walls are lost the robot should then return to wandering the environment.
4. **Turn on all 3 LEDs when the robot follows the center of the hallway.**

In the modified subsumption architecture, wander-randomly is on layer 0 with inputs of random direction and heading. Avoid-obstacle including collide and runaway behaviors is on layer 1 with an input of range sensors. Follow-wall is also on layer 1 and has an input of range sensors and it calls avoid-obstacle. Layer 2 is follow-hallway which has inputs of range sensors and subsumes the output from follow-wall.

The robot should wander until an obstacle is detected and attempt to navigate around it by maintaining a distance of 4 to 6 inches. If the robot encounters a wall or obstacles on both sides, it should move to the center of the two objects and move forward. You should attempt to address issues such as doors, getting unstuck from corners and turning corners (see Figure 5). Note that although the robot circumvents obstacle 1 in the figure, your architecture may cause the robot to get stuck in a loop circling the box. If this happens, what could you do to break the robot out of this endless loop? Devise a method to test and confirm that your program works correctly and present the results in the laboratory memo. Table 1 has a summary of the LED light sequence to indicate robot state.

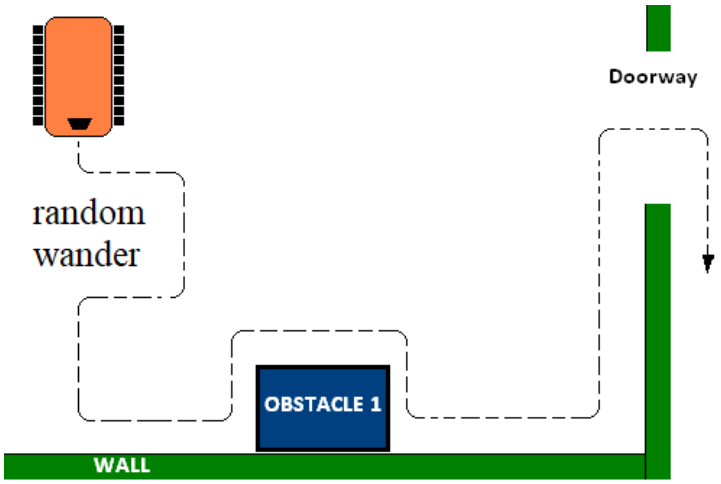

Figure 5: Wall Following Example

Table 1: LED state indicators

state	Red LED	Yellow LED	Green LED
Random wander			ON
Wall lost			
Too far	ON		
Too close		ON	
Right WF	ON	ON	

Left WF		ON	ON
Front wall	ON		ON
Follow center	ON	ON	ON

Part 5 – Go To Goal

Finally implement a go to goal behavior on the robot such that if it is given a goal position it will navigate around obstacles similar to walls to get from the initial to the final position.

Submission Requirements:

Software Design Plan

For each lab you will submit a software design plan which may be in the form of pseudocode, a flowchart, state machine or subsumption architecture. You will show this plan to the instructor during class, the plan will be graded and you will get feedback on implementation before designing the full system.

Demonstration:

Bring your robot fully charged to class every day! Plug it in overnight.

Similar to prior labs, the demonstration will involve showing that each behavior works separately and then that the integrated behaviors with the architecture works properly.

- To demonstrate the Follow-Wall behavior, the robot should be placed next to a wall and show that it can follow the wall at a distance between 4 and 6 inches for at least 2 to 3 feet on both the left and right side.
- The robot will also be tested on its ability to navigate an obstacle next to the wall and how it handles doorways in the wall (inner and outer corners).
- The next demonstration will be to place the robot in a hallway and show that it moves to the center and continues to follow the hallway until one or both walls are lost and then switch back to wall following.
- Next, the architecture will be evaluated by the robot starting in a wander behavior until an obstacle or wall is detected, the robot should then attempt to follow the object or wall at a distance of 4 to 6 inches unless a wall is detected on the opposite side. At that point the robot should attempt to follow the center of the hallway until one or both walls is lost.
- Finally, the robot will be given a goal position and have to navigate around an obstacle in the direct path in order to arrive at the goal point.

Bring your robot fully charged to class everyday.

Program:

The program should be properly commented and modular with each new behavior representing a new function call. The design of the subsumption architecture should be evident from the program layout.

Grading Rubric:

The lab is worth a total of 30 points and is graded by the following rubric.

Points	Demonstration	Code	Memo
100%	Excellent work, the robot performs exactly as required	Properly commented with a header and function comments, easy to follow with modular components	Follows all guidelines and answers all questions posed
75%	Performs most of the functionality with minor failures	Partial comments and/or not modular with objects	Does not answer some questions and/or has spelling, grammatical, content errors
50%	Performs some of the functionality but with major failures or parts missing	No comments, not modular, not easy to follow	Multiple grammatical, format, content, spelling errors, questions not answered
0	Meets none of the design specifications or not submitted	Not submitted	Not submitted

Upload Details:

You must submit your software design plan to GradeScope You must submit your properly commented code in the appendix of the lab worksheet to GradeScope. Check the course calendar for the lab demonstration due date.